Finite-State Machines with Weights

Algorithms for NLP

September 23, 2014
Note

• Often two ideas get introduced at the same time:
  1. weights (scores, probabilities)
  2. learning (estimation)
• We are going to talk about weights but not learning.
  – Learning algorithms use the algorithms we develop as subcomponents.
Weighted Automata

• Weighted automata can be understood as transducing strings to weights
• Weighted transducers can be understood as transducing pairs of strings to weights
# Weighted Finite-State Automaton

<table>
<thead>
<tr>
<th>Element</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q$</td>
<td>Finite set of states</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>Finite vocabulary</td>
</tr>
<tr>
<td>$I \subseteq Q$</td>
<td>Set of initial states</td>
</tr>
<tr>
<td>$F \subseteq Q$</td>
<td>Set of final states</td>
</tr>
<tr>
<td>$E \subseteq (Q \times (\Sigma \cup {\varepsilon}) \times Q)$</td>
<td>Set of transitions (edges)</td>
</tr>
<tr>
<td>$\lambda : I \rightarrow \mathbb{R}_{\geq 0}$</td>
<td>Initial weights</td>
</tr>
<tr>
<td>$\rho : F \rightarrow \mathbb{R}_{\geq 0}$</td>
<td>Final weights</td>
</tr>
<tr>
<td>$w : E \rightarrow \mathbb{R}_{\geq 0}$</td>
<td>Transition weights</td>
</tr>
</tbody>
</table>
Paths in WFSAs

- Define the function \( p(e) \) to be the previous state of an edge \( e \) in \( E \)
- Define the function \( n(e) \) to pick out the next state of an edge \( e \)
- A path \( \pi \) in \( E^* \) is a sequence of transitions \( e_1 e_2 \ldots e_\ell \) such that \( n(e_i) = p(e_{i+1}) \) for all \( i \in [1, \ell) \)
- We overload \( p \) and \( n \) to be defined on paths:
  \[ p(\pi) = p(e_1) \quad n(\pi) = p(e_\ell) \]
Weight of a Path

• We generalize the transition weight of the path as the **product** of all transitions

\[
    w(\pi) = w(e_1) \times w(e_2) \times \cdots \times w(e_\ell)
\]

\[
    = \prod_{i=1}^{\ell} w(e_i)
\]

• The **output weight (or score)** of the path is defined to be

\[
    \text{score}(\pi) = \lambda(\rho(\pi)) \times w(\pi) \times \rho(n(\pi))
\]
Deterministic and Nondeterministic

- Deterministic WFSA
  - There is one initial state $q_0$
  - For each $q$ in $Q$ and $s$ in $\Sigma$, there is at most one $r$ in $Q$ such that $(q, s, r) \in E$.
  - There is no $(q, r) \in (Q \times Q)$ s.t. $(q, \varepsilon, r) \in E$

- Can we determinize WFSAs?
  - Sometimes, but not always

- Can we minimize (deterministic) WFSAs?
  - Yes. There might be more than one way to define “minimization.”
Determinism

• Let $P(q,r)$ designate the set of paths from $q$ to $r$.
  – Generalize this definition to sets $Q,R$ in the usual way.

• Let $P(q,x,r)$ designate the set of paths from $q$ to $r$ producing the sequence $x$.
  – Generalize this definition to sets $Q,R$ in the usual way.
Weighted Determinization

- Multiple paths can produce the same output
- In weighted determinization these path weights must be aggregated
- How do we aggregate?
  - Sum
  - Max
  - Others?
A WFSA You Can't Determine

- Union of two $a^*$ WFSA's.
- Want weight of $a^n$ to be $0.5^n + 0.3^n$ (why plus?)
- Deterministic WFSA accepting $a^n$ must have a "lollipop" topology ...
Structure of a Deterministic WFSA for $a^*$

- Let $n = j + kl$
- Weight of $a^n$ will be $pq^k r$, which decays geometrically

$j$ is the length of the stem; total weight is $p$

$l$ is the circumference of the “head”; total weight is $q$ for going around it
Ambiguous and Unambiguous

• Unambiguous WFSA:
  – There is at most one accepting path per string.

• Can we determinize unambiguous WFSA?
  – Sometimes, but not always.
Why Weights?

• In a DFSA, weights let us define a mapping from strings to real numbers (scores).
• In an NDFSA, weights let us compare different paths.
  – Better property to think about: *ambiguous* WFSA.
An Unambiguous WFSA You Can't Determinize

*Thanks to Jason Eisner for the example.*
Road Map

- Application of deterministic WFSAs: language models
- Algorithms for best paths in ambiguous WFSAs
- WFSTs in brief
Language Models

• Mapping of sequences to (nonnegative) scores.
  – Usually this number is interpreted as a probability.
• Mapping a sequence to zero means it is not allowed.
• LMs are central to machine translation and speech recognition, where they are often used in a noisy channel framework.
Noisy Channel Paradigm

probability distribution over strings (language model)

source

what we want ("plaintext")

channel

probability distribution over observations given strings (acoustic or translation model)

what we see ("ciphertext")

decode

details of the decoding algorithm depend on details of the source and channel models
The Simplest Language Model

<table>
<thead>
<tr>
<th>word</th>
<th>transition weight ($w$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>,</td>
<td>0.05</td>
</tr>
<tr>
<td>the</td>
<td>0.04</td>
</tr>
<tr>
<td>.</td>
<td>0.04</td>
</tr>
<tr>
<td>of</td>
<td>0.02</td>
</tr>
<tr>
<td>to</td>
<td>0.02</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Friday</td>
<td>0.002</td>
</tr>
</tbody>
</table>

$\lambda = 1$

$\rho = 0.04$
What Else Might We Do?

• We can encode more information in the states.
  – More states: more information.
  – Often used to encode history and increase the “memory” of the process.
  – Example: create a distinct state for each (n-1) word history ... this is an n-gram language model.
A Simple Bigram Model on \{a, b, c\}
What Else Might We Do?

• We can encode more information in the states.
  – More states: more information.
  – Often used to encode history and increase the “memory” of the process.
    – Example: create a distinct state for each (n-1) word history ... this is an **n-gram** language model.

• We can take a random walk through the network to generate text.
  – Not very useful, just interesting.
For Fun

• Unigram model estimated on 2.8M words of American political blog text.
For Fun

• Bigram model estimated on 2.8M words of American political blog text.

the lack of the senator mccain hadn t keep this story backwards
while showering praise of the kind of gop weakness
it was mistaken for american economist anywhere in the
white house press hounded the absence of those he s
as a wide variety of this election day after the
candidate b richardson was polled ri in hempstead
moderated by the convention that he had zero wall
street journal argues sounds like you may be the
primary
but even close the bill told c e to take the obama on
the public schools and romney
fred flinstone s see how a lick skillet road it s
little sexist remarks
For Fun

- Trigram model estimated on 2.8M words of American political blog text.

as i can pin them all none of them want to bet that any of the might be conservatism unleashed into the privacy rule book and when told about what paul fans organized another massive fundraising initiative yesterday and i don't know what the rams supposedly want ooh but she did but still victory dinner alone among republicans there are probably best not all of the fundamentalist community asked for an independent maverick now for crystallizing in one especially embarrassing
he realizes fully how shallow and insincere conservative behavior has been he realizes that there is little way to change the situation this recent arianna huffington item about mccain issuing heartfelt denials of things that were actually true or for that matter about the shi a sunni split and which side iran was on would get confused about this any more than someone with any knowledge of us politics would get confused about whether neo confederates were likely to be supporting the socialist workers party at the end of the world and i m not especially discouraged now that newsweek shows obama leading by three now
What Else Might We Do?

• We can encode more information in the states.
  – More states: more information.
  – Often used to encode history and increase the “memory” of the process.
  – Example: create a distinct state for each (n-1) word history ... this is an **n-gram** language model.

• We can take a random walk through the network to generate text.
  – Not very useful, just interesting.

• We can calculate the score of a sequence ... 
  – How?
Scoring a Sequence with a Deterministic WFSA

**Input:** WFSA, $s \in \Sigma^*$

$q := q_0$ such that $l = \{q_0\}$

$a := \lambda(q_0)$

**for** $t = 1$ to $|s|$:  

$N = (\{q\} \times \{s_t\} \times Q) \cap E$

if $N = \emptyset$ return 0

if $|N| > 1$ return *nondeterministic*

$e := e'$ such that $N = \{e'\}$

$a := a \times w(e)$

$q := n(e)$

if $q \notin F$ return 0

return $a \times \rho(q)$
Additive Formulation

• We can think of the weights as *additive* rather than *multiplicatively*.
  – Score of a path is the *sum* of its transitions' weights.
  – If you like probabilities, think of these as log-probabilities.

• Weights greater than zero are like “bonuses.”

• Weights less than zero are like penalties.
General Case: Ambiguous WFSAs
Ambiguous WFSAs

• A string may have more than one state sequence.
• Useful for representing ... ambiguity!
  – Weights define relative goodness of different paths.
• Recall from last time: the *path* can be used to encode an analysis.
  – E.g., if symbols are words, the state before or after might correspond to the word's part of speech.
  – More later.
Toward an Algorithm

- We can use weights to encourage or discourage paths.
- In an ambiguous FSA, there might be multiple paths for the same string.
- Important algorithm: find the *best* path for a string.
A Tempting, Incorrect Algorithm
Tempting but Incorrect

- Pick the “best” start state.

\[ q_0 := \arg \max_{q \in I} \lambda(q) \]

- For \( t = 1 \) to \(|s|\):

\[ q_t := \arg \max_{q \in Q} w(q_{t-1}, s_t, q) \]
Why It’s Wrong In General

baz
Problems

• What if there is no transition out of $q_{t-1}$ that matches $s_t$?

• If there are any epsilons in the sequence, *we can't see them*!
  – This algorithm will never take an $\epsilon$-edge.

• Something fishy about leaving out the stopping weights $\rho$!

• In general: a seemingly good decision early on could turn out to be very bad later on, since rewards and punishments can always be delayed until the very end.
ε-Transitions

• Let’s continue to ignore them
  – There is a weighted “remove epsilons” operation that preserves best paths.

• We will come back to this briefly.
An Easier Problem

- If I knew the best path $\hat{e}_1 \cdots \hat{e}_{\ell-1}$ for the first $\ell-1$ symbols $s_1 \ldots s_{\ell-1}$, then the decision for the final part would be easy:

$$q_\ell := \arg \max_{q \in F} w(n(\hat{e}_{\ell-1}), s_\ell, q) \times \rho(q)$$

- Note that this choice only depends on $q_{\ell-1}$, not the whole path prefix $q_1 \ldots q_{\ell-1}$.
- Manageable: best path prefix for $s_1 \ldots s_{\ell-1}$ ending in each $q$. 
Toward a Recurrence

• Let $\alpha(q, i)$ be the score of the best path for the length $t$ prefix $(s_1...s_t)$ ending in state $q$.

\[
\alpha(q, t) = \max_{e_1e_2...e_t:n(e_t)=q} \lambda(p(e_1)) \times \prod_{i=1}^{t} w(e_i) \quad \forall q \in Q, \forall t \in [1, \ell)
\]

\[
= \max_{e_1e_2...e_t:n(e_t)=q} \lambda(p(e_1)) \times \left( \prod_{i=1}^{t-1} w(e_i) \right) \times w(\langle n(e_{t-1}), s_t, q \rangle)
\]

\[
= \alpha(q_{t-1}, t-1)
\]

\[
= \max_{q_{t-1} \in Q} \alpha(q_{t-1}, t - 1) \times w(\langle q_{t-1}, s_t, q \rangle)
\]
Recovering the Best Path

• We are given the prefix best-path scores:
  \[ \alpha : Q \times \{0, 1, \ldots, \ell\} \rightarrow \mathbb{R}_{\geq 0} \]

\[ q_\ell := \arg \max_q \alpha(q, \ell) \times \rho(q) \]

for \( t = \ell - 1 \) to 0

\[ q_t := \arg \max_q \alpha(q, t) \times w(q, s_{t+1}, q_{t+1}) \]
\[ \alpha(q, i) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>(\ell)-2</th>
<th>(\ell)-1</th>
<th>(\ell)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q_0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(q_1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(q_2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(q_3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(q_4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(q_5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(q_6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\( \alpha(q, i) \)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>( \ell-2 )</th>
<th>( \ell-1 )</th>
<th>( \ell )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( \lambda(q_0) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( \lambda(q_1) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( \lambda(q_2) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( \lambda(q_3) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>( \lambda(q_4) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_5 )</td>
<td>( \lambda(q_5) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_6 )</td>
<td>( \lambda(q_6) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \forall q \in I, \quad \alpha(q, 0) \leftarrow \lambda(q) \]
$\alpha(q, i)$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>$\ell$-2</th>
<th>$\ell$-1</th>
<th>$\ell$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$\lambda(q_0)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_1$</td>
<td>$\lambda(q_1)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td>$\lambda(q_2)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>$\lambda(q_3)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_4$</td>
<td>$\lambda(q_4)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_5$</td>
<td>$\lambda(q_5)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_6$</td>
<td>$\lambda(q_6)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\forall q \in Q, \quad \alpha(q, 1) = \max_{e \in (Q \times \{s_1\} \times \{q\} \cap E)} \alpha(p(e), 0) \times w(e)$
\[ \forall q \in Q, \quad \alpha(q, 2) = \max_{e \in (Q \times \{s_2\} \times \{q\} \cap E)} \alpha(p(e), 1) \times w(e) \]
\( \alpha(q, i) \)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>( \ell-2 )</th>
<th>( \ell-1 )</th>
<th>( \ell )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( \lambda(q_0) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( \lambda(q_1) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( \lambda(q_2) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( \lambda(q_3) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>( \lambda(q_4) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_5 )</td>
<td>( \lambda(q_5) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_6 )</td>
<td>( \lambda(q_6) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( \forall q \in Q, \forall i \in [1, \ell] \quad \alpha(q, t) = \max_{e \in (Q \times \{s_t\} \times \{q\} \cap E)} \alpha(p(e), t-1) \times w(e) \)
Constructing Prefix Best-Path Scores

Input string is \((s_1 \ s_2 \ldots \ s_\ell)\)
Goal: construct \(\alpha : Q \times \{0, 1, \ldots, \ell\} \rightarrow \mathbb{R}_{\geq 0}\)

\[
\text{set all } \alpha(q, i) := 0 \\
\text{for each } q \in I: \ \alpha(q, 0) := \lambda(q) \\
\text{for } t = 1 \text{ to } \ell:\ \\
\quad \text{for each } q \in Q:\ \\
\quad \quad \text{for each } q' \in Q:\ \\
\quad \quad \quad \alpha(q, t) := \max\{ \alpha(q, t), \ \alpha(q', t-1) \times w(q', s_t, q) \} 
\]
High-Level Best WFSA Path Algorithm

• First, construct the prefix best-path scores $\alpha$.
• Then recover the path.

• Alternative: maintain pointers to each “argmax” when constructing $\alpha$, then follow the pointers to recover the path.

• Runtime and space requirements for this algorithm, in $|Q|$ and in $\ell$?
Handling $\varepsilon$ Transitions

initialize all $\alpha(q, i)$ to 0
for each $q$: $\alpha(q, 0) := \pi(q)$
for $i = 1$ to $n$:
  for each $q$:
    for each $q'$:
      $\alpha(q, i) := \max\{ \alpha(q, i), \alpha(q', i-1) \times \delta(q', s_i, q) \}$
repeat until $\alpha(\ast, i)$ converge:
  for each $q$:
    for each $q'$:
      $\alpha(q, i) := \max\{ \alpha(q, i), \alpha(q', i) \times \delta(q', \varepsilon, q) \}$
Problems with $\varepsilon$

• Runtime is much harder to analyze, because of $\varepsilon$ cycles.

• Two cases:
  
  – Dampening cycles, where hypothesizing lots of $\varepsilon$ transitions makes the path's score worse. Example: $\delta(q, \varepsilon, q) < 1$
  
  – Amplifying cycles, e.g., $\delta(q, \varepsilon, q) > 1$. 

WFSA
weighted languages disambiguation

FSA
regular languages

FST
regular relations string-to-string mappings
WFSA
weighted languages disambiguation

WFST
weighted string-to-string mappings

FSA
regular languages

regular relations
string-to-string mappings
Beyond WFSAs: WFSTs

• Weighted finite-state *transducers*: combine the two-tape idea from last time with the weighted idea from today.
  – Very general framework; key operation is **weighted composition**.
  – Best-path algorithm variations: best path and output given input, best path given input and output, ...

• Weights do not have to be numbers.
  – Boolean weights = traditional FSAs/FSTs.
  – Real weights = the case we explored so far.
  – Many other *semirings* (sets of possible weight values and operations for combining weights)
WFSAs and WFSTs: Algorithms

• OpenFST documentation contains high-level view of most algorithms you think you want, and pointers.
• The set of people developing general WFST algorithms largely overlaps with the OpenFST team.
  – Mehryar Mohri, Cyril Allauzen, Michael Reilly
• This is an active area of research!